Structure-preserving Graph Neural Networks

Prof. O-Joun Lee

Dept. of Artificial Intelligence,
The Catholic University of Korea
ojlee@catholic.ac.kr

네트워크 과학 연구실
NETWORK SCIENCE LAB

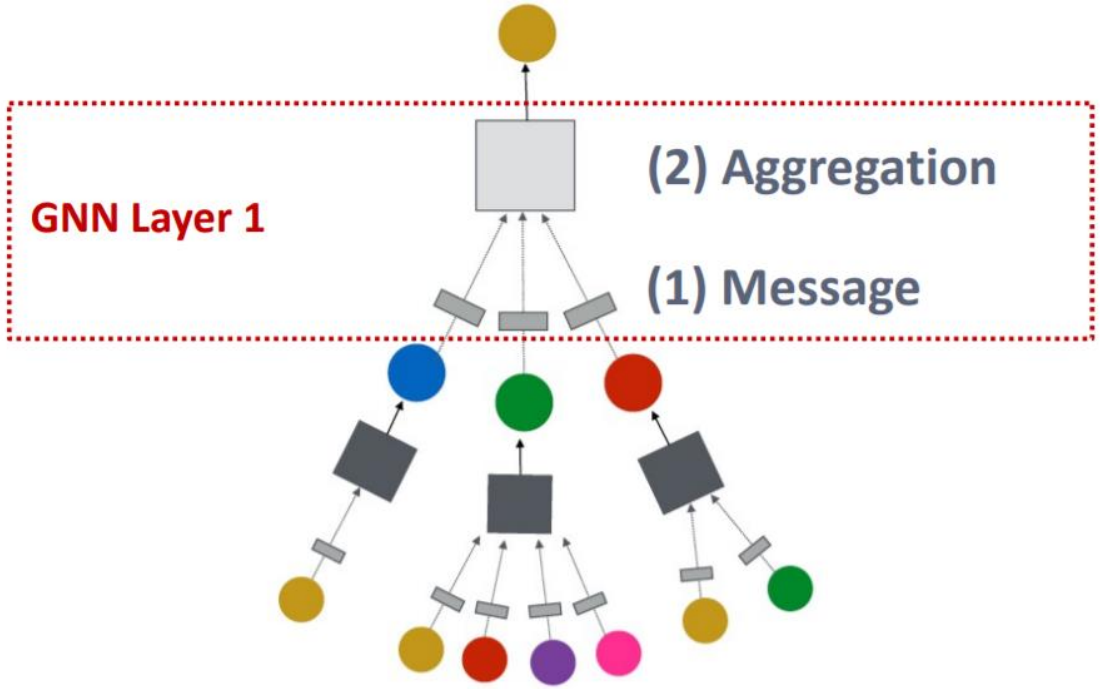
가톨릭대학교
THE CATHOLIC UNIVERSITY OF KOREA

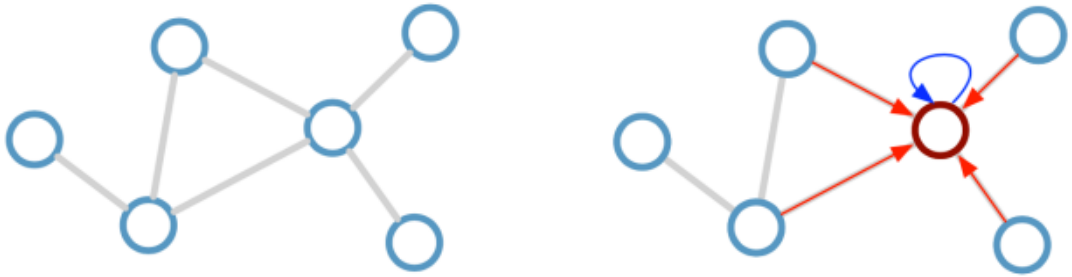
Contents

- Computational graph
- Graph Isomorphism and GNN problems
- Expressive power of GNNs
- How to build the most powerful GNNs?

Recap: Graph Convolutional Networks (GCNs)

- GNN Layer = **Message** + **Aggregation**
  - Message COMPUTATION
    - how to make each neighborhood node as embedding?
  - Message AGGERGATION
    - how to combine those embeddings?





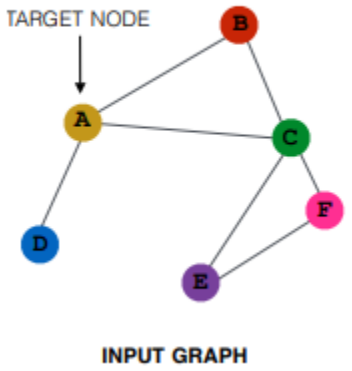
Update rule: $$\mathbf{h}_i^{(l+1)} = \sigma\left(\mathbf{h}_i^{(l)}\mathbf{W}_0^{(l)} + \sum_{j\in\mathcal{N}_i}\frac{1}{c_{ij}}\mathbf{h}_j^{(l)}\mathbf{W}_1^{(l)}\right)$$

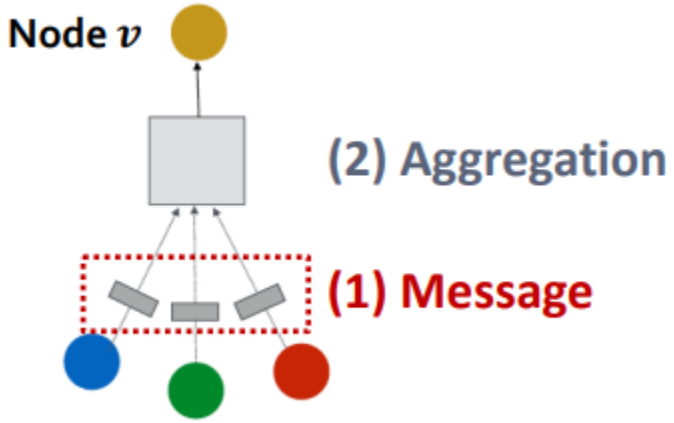
Kipf & Welling (ICLR 2017), related previous works by Duvenaud et al. (NIPS 2015) and Li et al. (ICLR 2016)

GNN Layer: Message Computation

- Intuition: Each node will create a message, which will be sent to other nodes later
- Example: A Linear layer $\mathbf{m}_u^{(l)} = \mathbf{W}^{(l)}\mathbf{h}_u^{(l-1)}$
  - Multiply node features with weight matrix $\mathbf{W}^{(l)}$

Message function: $\mathbf{m}_u^{(l)} = \mathrm{MSG}^{(l)}\left(\mathbf{h}_u^{(l-1)}\right)$





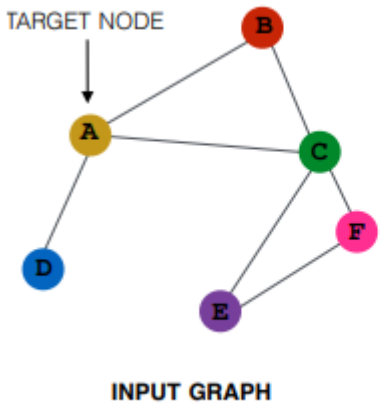
GNN Layer: Message Aggregation

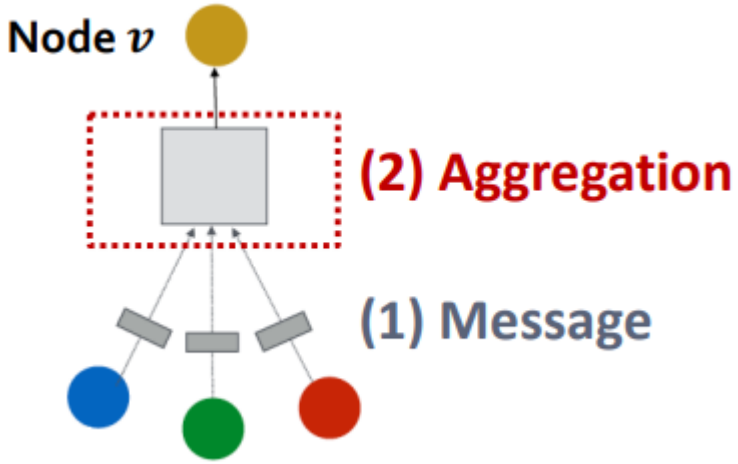
- Intuition: Each node will aggregate the messages from node v's neighbors

$$\mathbf{h}_v^{(l)} = \mathrm{AGG}^{(l)}\left(\left\{\mathbf{m}_u^{(l)}, u \in N(v)\right\}\right)$$

- Example: Sum(·), Mean(·) or Max(·) aggregator

$$\mathbf{h}_v^{(l)} = \mathrm{Sum}(\{\mathbf{m}_u^{(l)}, u \in N(v)\})$$





GNN Layer: Message Aggregation Issue

- **Issue**: Information from node v itself could get lost
  - Computation of $\mathbf{h}_v^{(l)}$ oes not directly depend on $\mathbf{h}_v^{(l-1)}$
- **Solution**: Include $\mathbf{h}_v^{(l-1)}$ when computing $\mathbf{h}_v^{(l)}$
  - (1) Message: compute message from node v itself

$$\mathbf{m}_u^{(l)} = \mathbf{W}^{(l)}\mathbf{h}_u^{(l-1)} \qquad \mathbf{m}_v^{(l)} = \mathbf{B}^{(l)}\mathbf{h}_v^{(l-1)}$$

  - (2) Aggregation: After aggregating from neighbors, we can aggregate the message from node v itself
    - Via concatenation or summation

Then aggregate from node itself

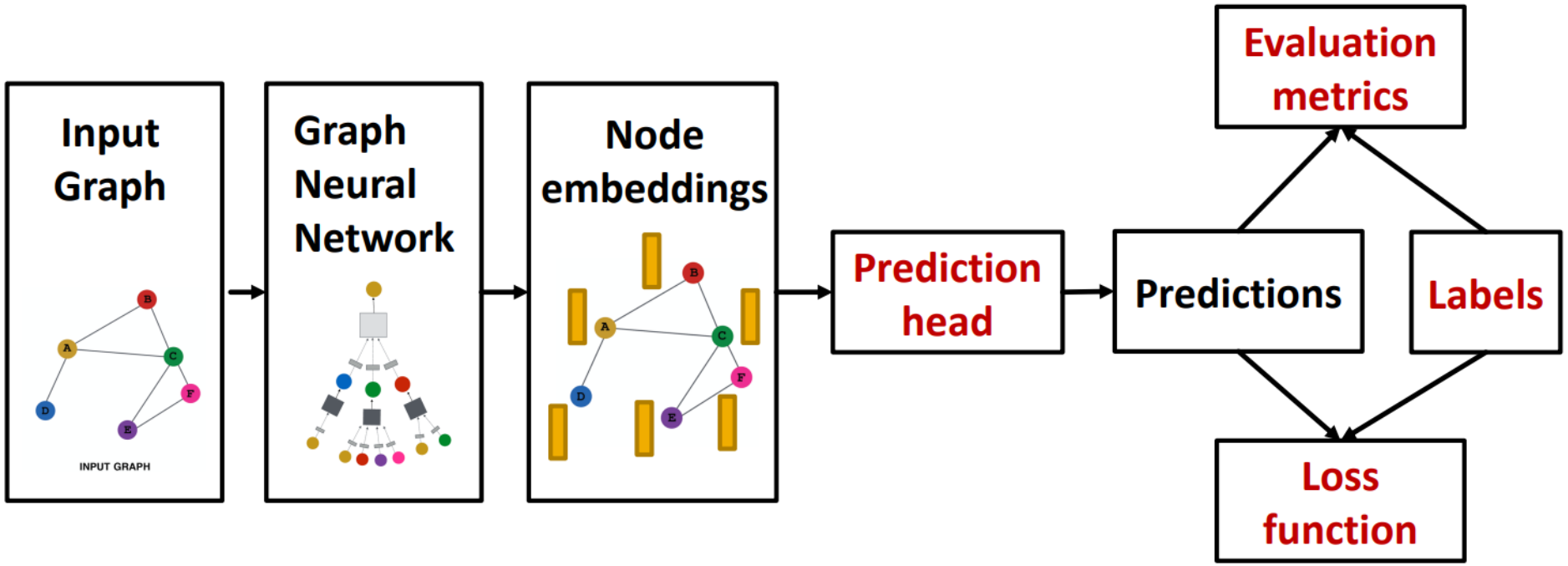
$$\mathbf{h}_v^{(l)} = \text{CONCAT}\left(\text{AGG}\left(\left\{\mathbf{m}_u^{(l)}, u \in N(v)\right\}\right), \mathbf{m}_v^{(l)}\right)$$

First aggregate from neighbors

https://web.stanford.edu/class/cs224w/slides/04-GNN2.pdf

Recap: GNN training pipeline

- Inputs: a graph dataset
- Outputs: node embeddings

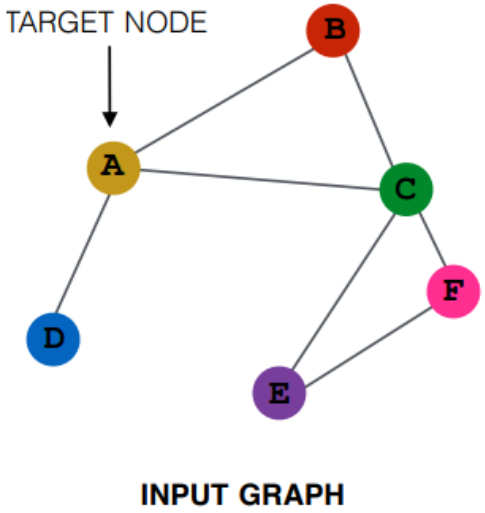


Expressive Power of GNNs

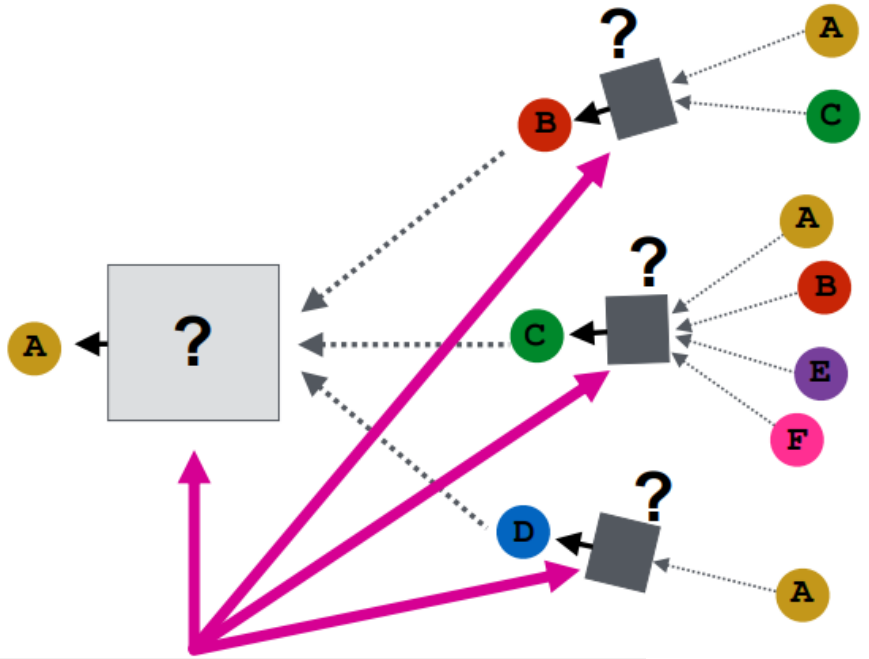
- How powerful are GNNs?
  - Many GNN models have been proposed (e.g., GCN, GAT, GraphSAGE, design space).
  - What is the expressive power (ability to distinguish different graph structures) of these GNN models?
  - How to design a maximally expressive GNN model?

Background: Many GNN models

- Many GNN models have been proposed:
  - GCN, GraphSAGE, GAT, Design Space etc.



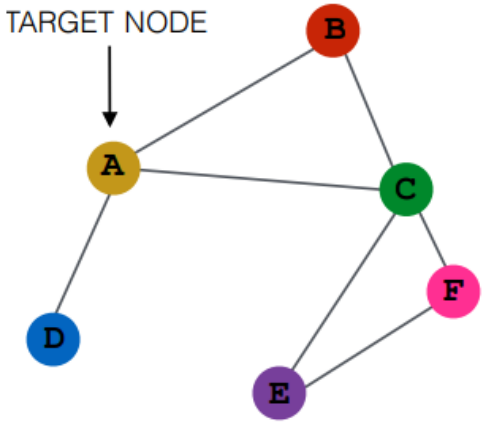
INPUT GRAPH



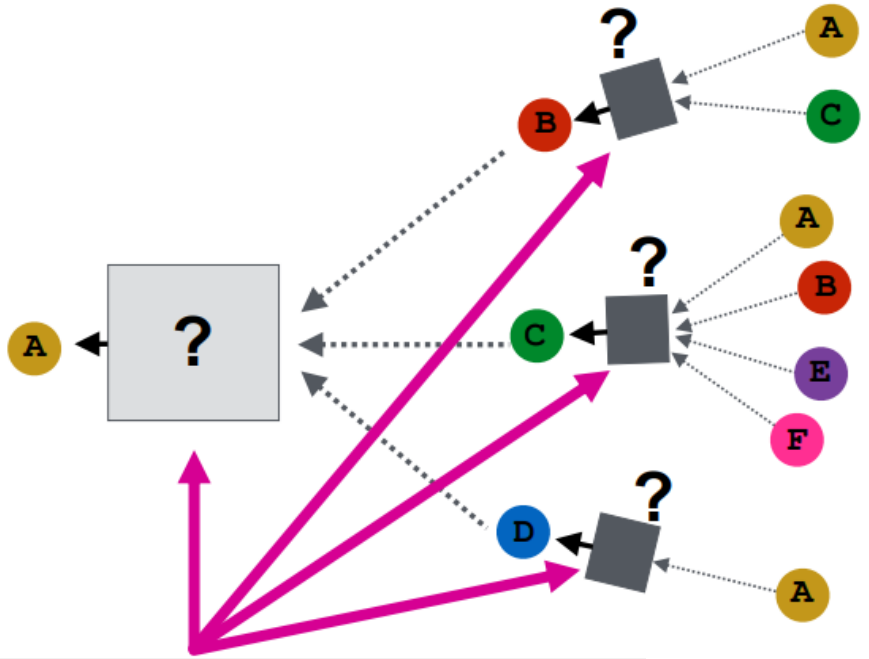
Different GNN models use different neural networks in the box

Background: Many GNN models

- GCN (mean-pool) [Kipf and Welling ICLR 2017]



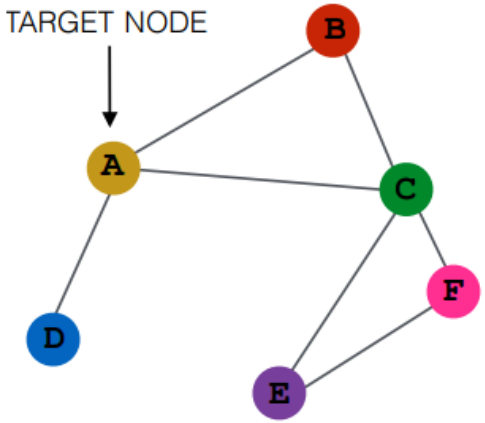
INPUT GRAPH



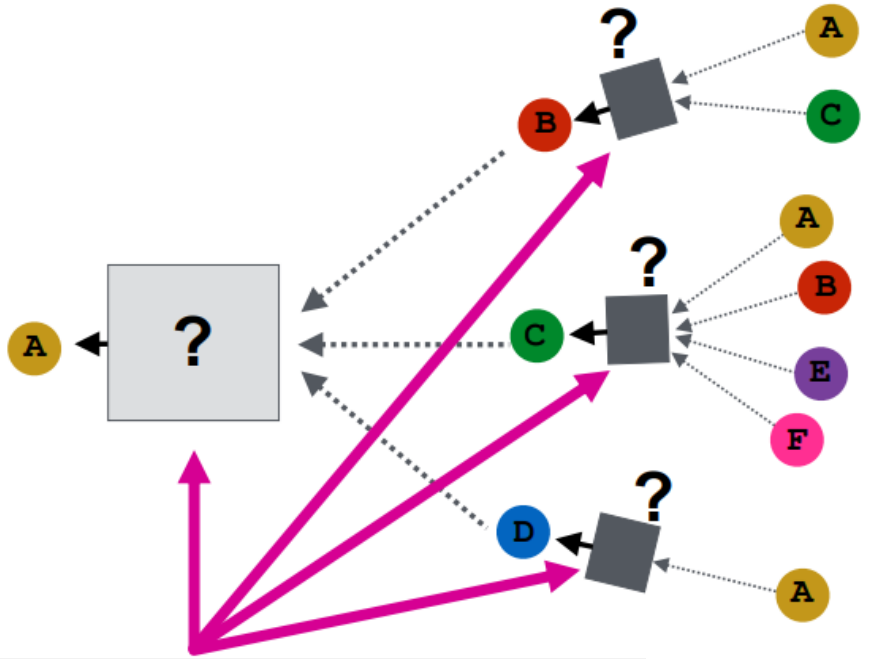
Element-wise mean pooling + Linear + ReLU non-linearity

Background: Many GNN models

- GraphSAGE (max-pool) [Hamilton et al. NeurIPS 2017]

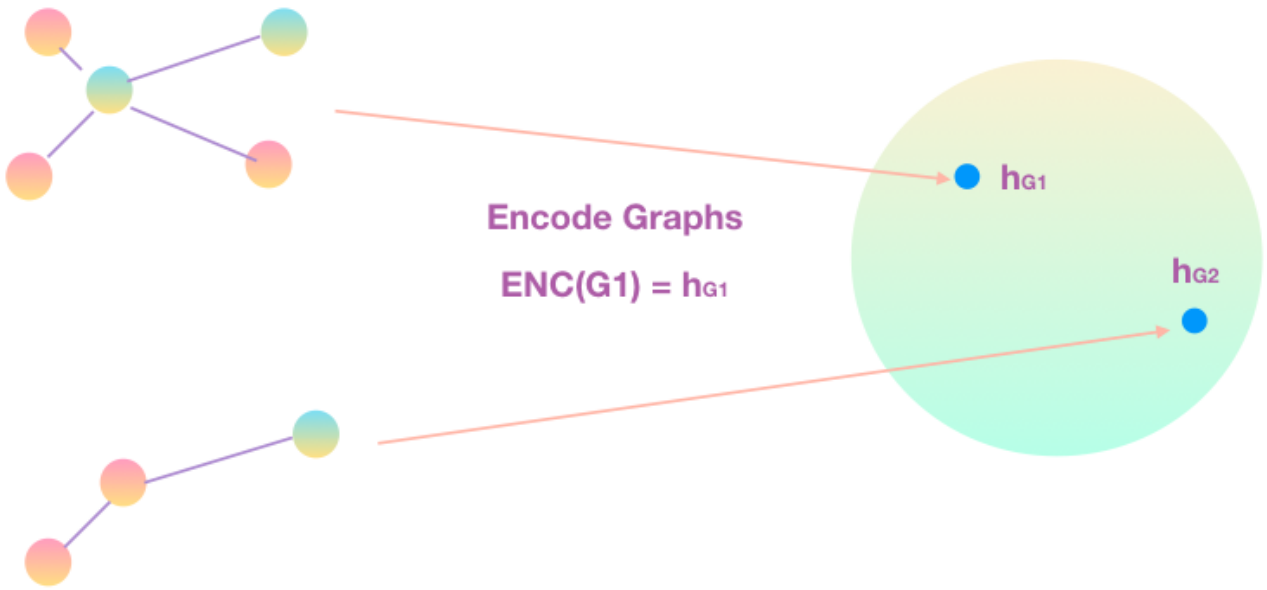


INPUT GRAPH



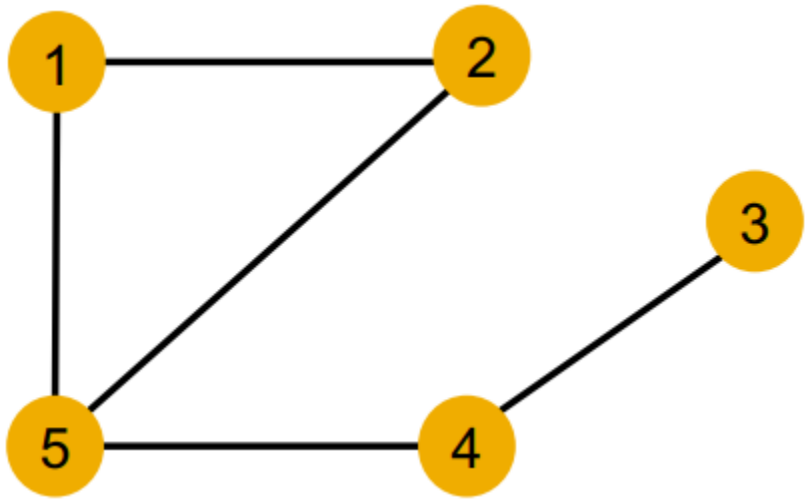
MLP + element-wise max-pooling

- **Do similar graphs really have the same label?**
  - The Graph Classification problem and the Graph Isomorphism problem are not the same.
  - Graph Isomorphism can be judged as isomorphic if the positions on the vector are close, but Graph Classification adds an element called "Label" to the graph, so that it is classified by the same Label, not by position.



Recap: Node colouring problem

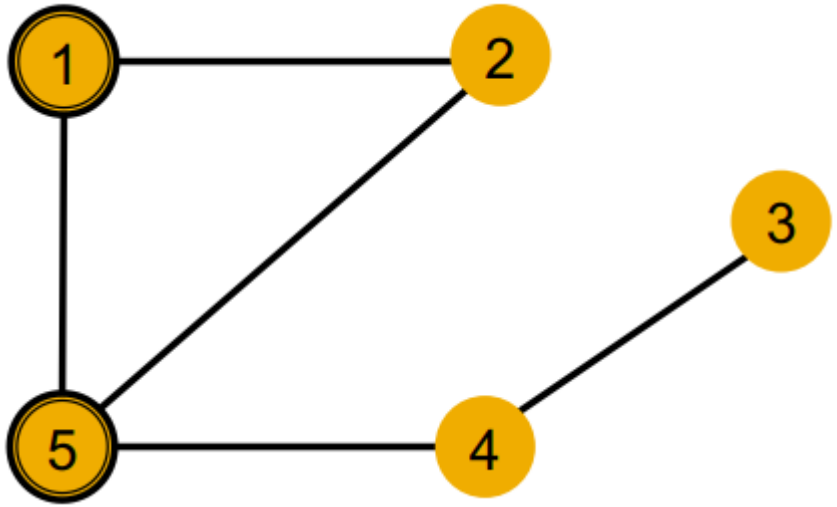
- We use node same/different colors to represent nodes with same/different features.
  - For example, the graph below assumes all the nodes share the same feature



Key question: How well can a GNN distinguish different graph structures?

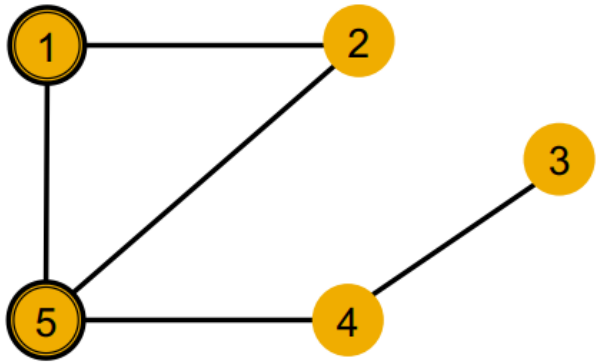
How well can a GNN distinguish different graph structures?

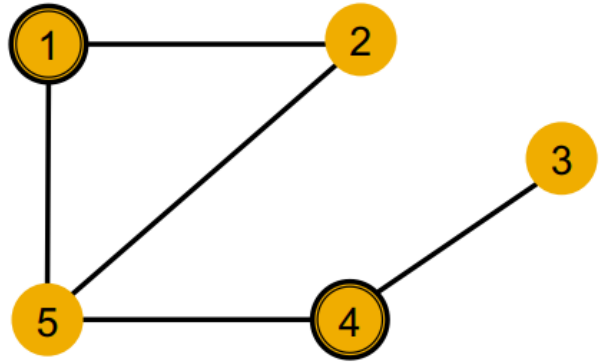
- **Local Neighbourhood structures:**
  - We specifically consider **local neighborhood structures** around each node in a graph.
  - Example: Nodes 1 and 5 have different neighborhood structures because they have different node degrees.



How well can a GNN distinguish different graph structures?

- **Local Neighbourhood structures:**
  - We specifically consider **local neighborhood structures** around each node in a graph.
  - Example: Nodes 1 and 5 have different neighborhood structures because they have different node degrees.
  - Nodes 1 and 4 both have the same node degree of 2. However, they still have different neighborhood structures because their neighbors have different node degrees.

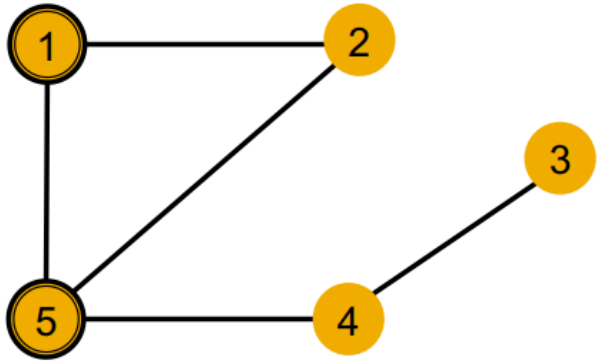


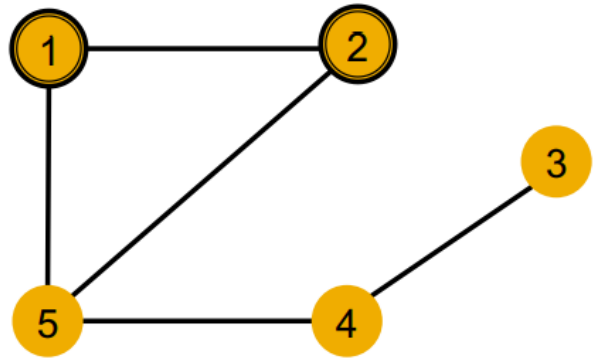


Node 1 has neighbors of degrees 2 and 3.
Node 4 has neighbors of degrees 1 and 3.

How well can a GNN distinguish different graph structures?

- **Local Neighbourhood structures:**
  - We specifically consider **local neighborhood structures** around each node in a graph.
  - Example: Nodes 1 and 5 have different neighborhood structures because they have different node degrees.
  - Nodes 1 and 4 both have the same node degree of 2. However, they still have different neighborhood structures because their neighbors have different node degrees.





Node 1 has neighbors of degrees 2 and 3.
Node 2 has neighbors of degrees 2 and 3.
And even if we go a step deeper to 2nd hop neighbors, both nodes have the same degrees (Node 4 of degree 2)

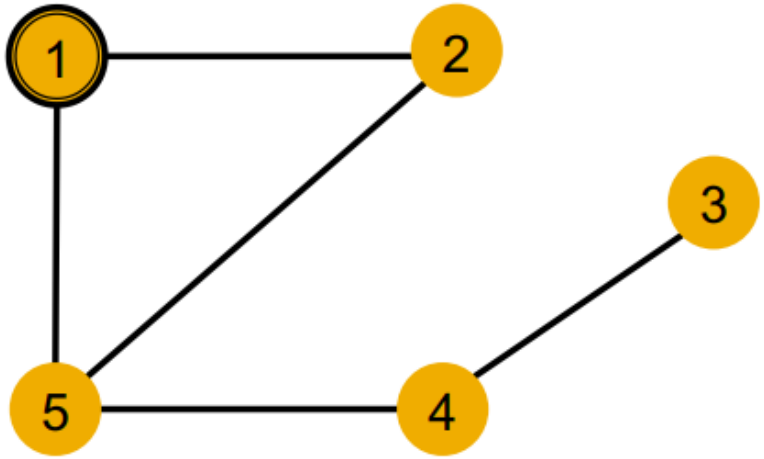
Local neighborhood structures

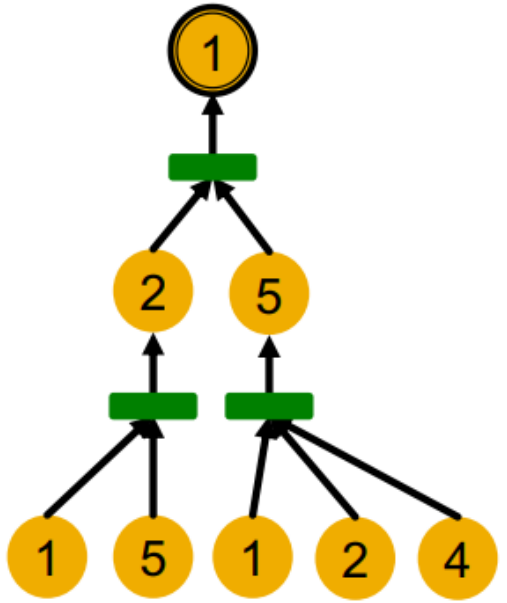
- **Key question:** Can GNN node embeddings distinguish different node's local neighborhood structures?
  - If so, when? If not, when will a GNN fail?

- We need to understand how a GNN captures local neighborhood structures.
  - **Key concept:** Computational graph

Computational graph

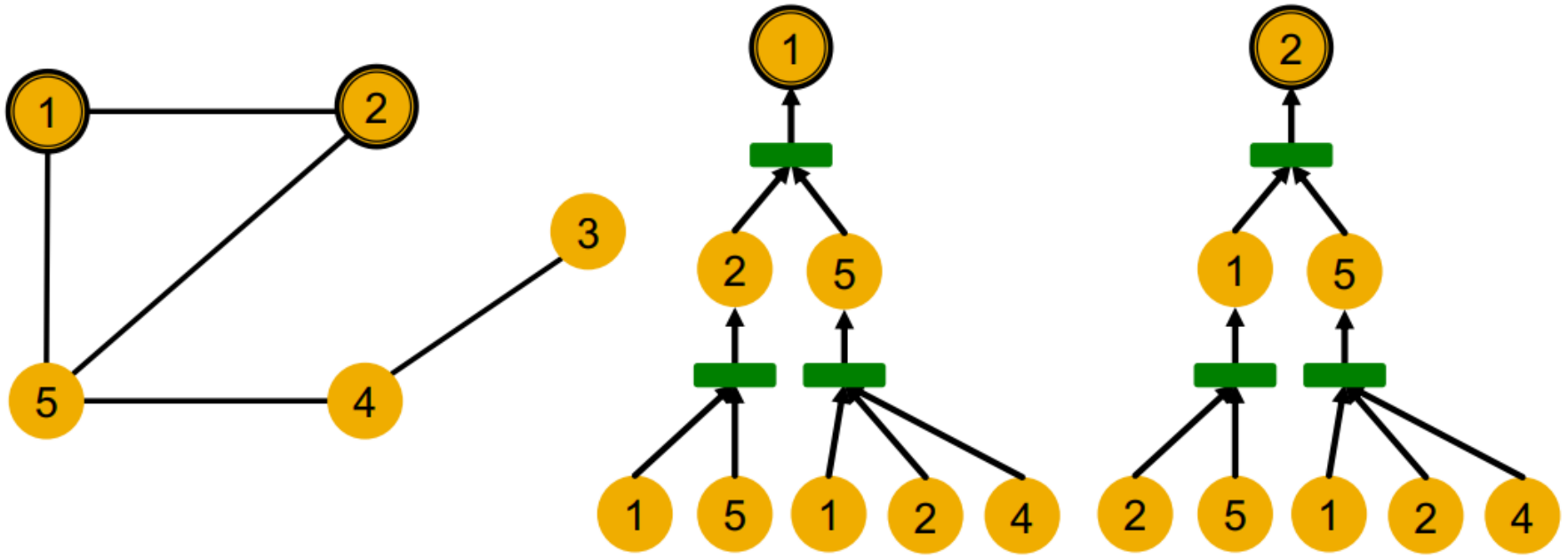
- In each layer, a GNN aggregates neighboring node embeddings.
- A GNN generates node embeddings through a computational graph defined by the neighborhood.
  - Ex: Node 1's computational graph (2-layer GNN)





Computational graph

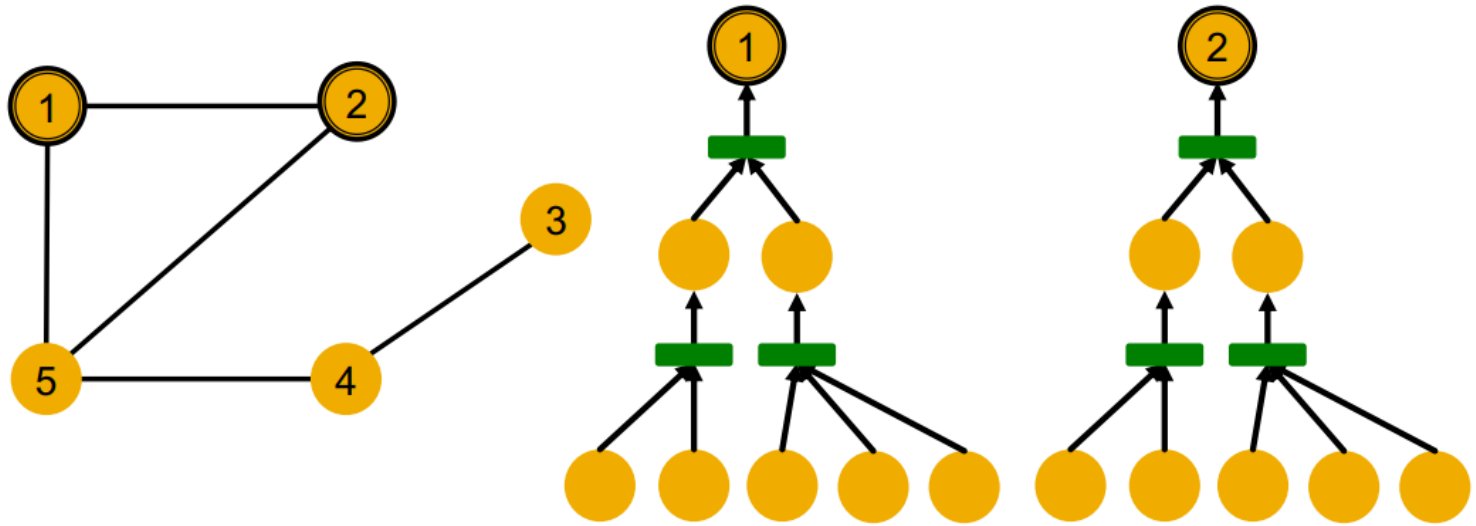
- In each layer, a GNN aggregates neighboring node embeddings.
- A GNN generates node embeddings through a computational graph defined by the neighborhood.
  - Ex: Nodes 1 and 2's computational graphs.



But GNN only sees node features (not IDs)

- A GNN will generate the same embedding for nodes 1 and 2 because:
  - Computational graphs are the same.
  - Node features (colors) are identical.

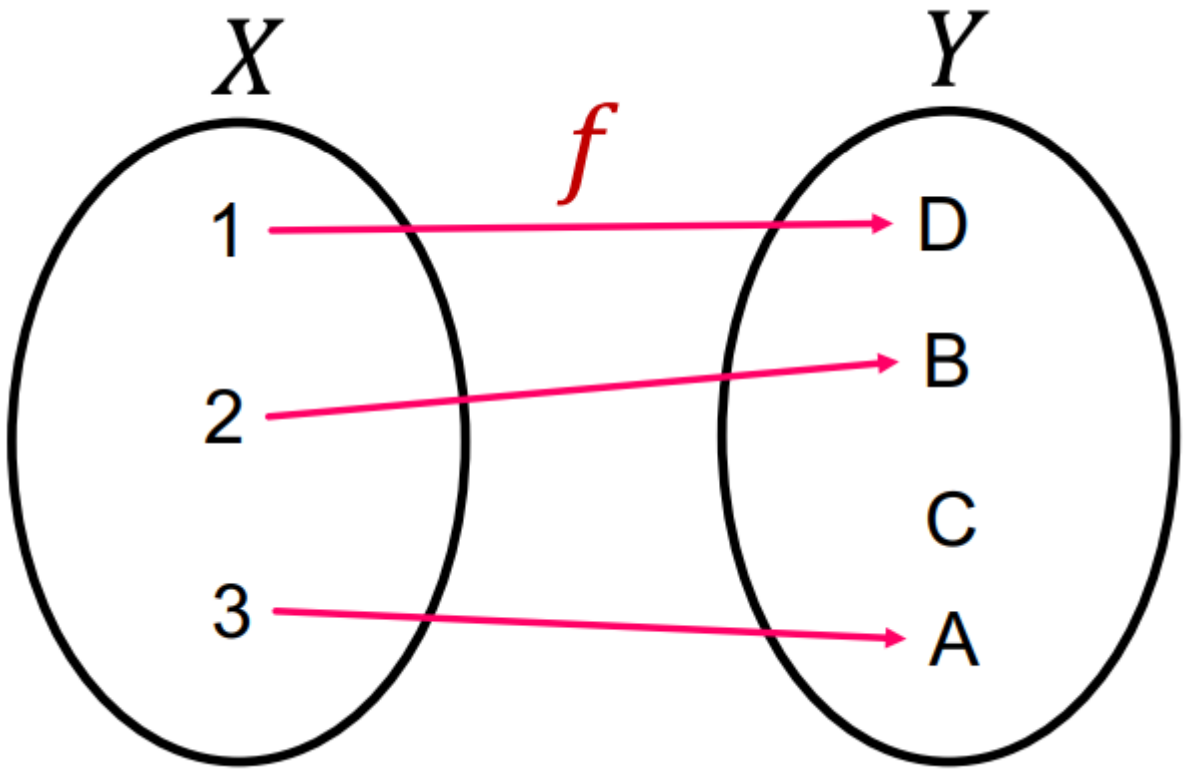
Note: GNN does not care about node ids, it just aggregates features vectors of different nodes.



GNN won't be able to distinguish nodes 1 and 2

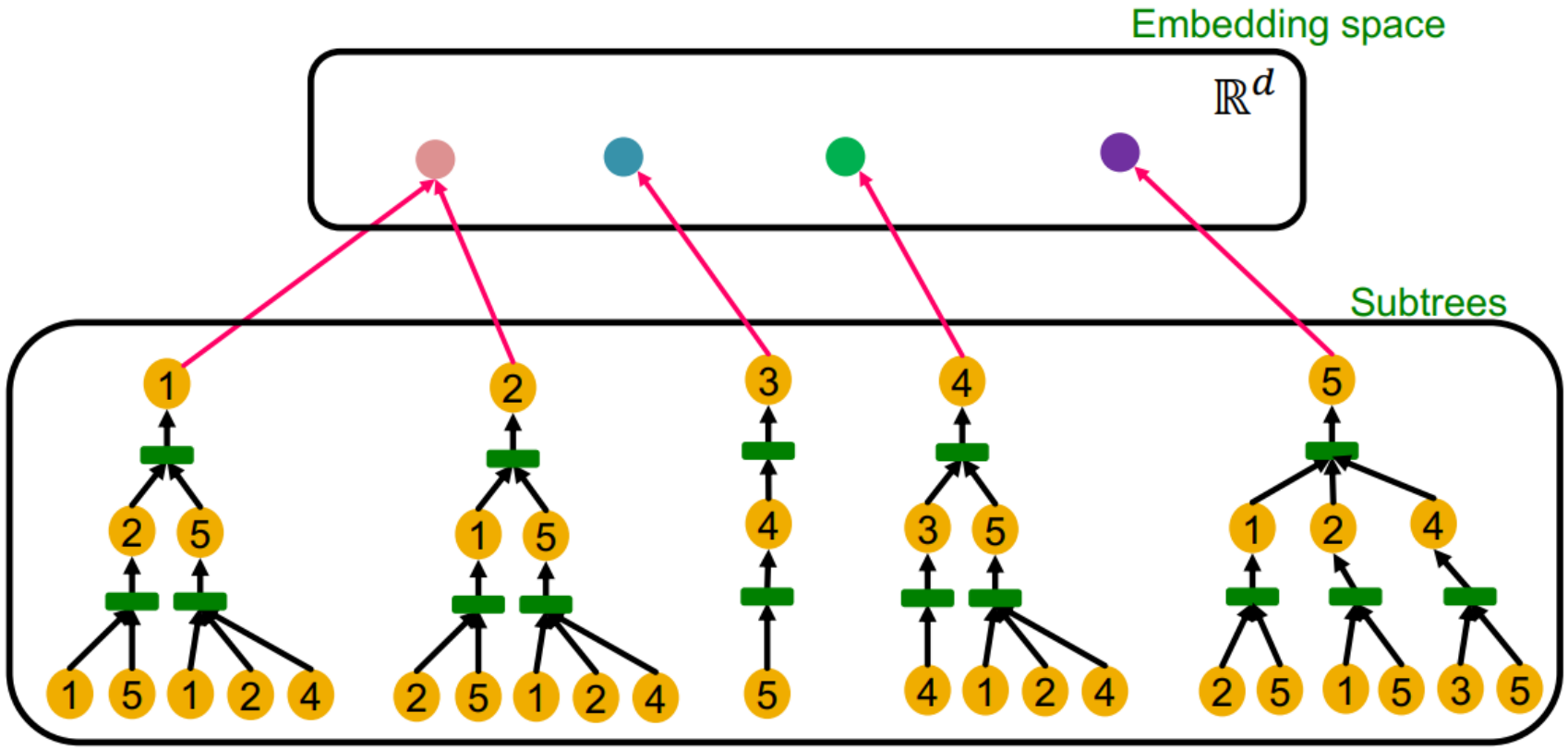
Recall: Injective funtion

- Function $f: X \rightarrow Y$ is injective if it maps different elements into different outputs.
- Intuition: f retains all the information about input.



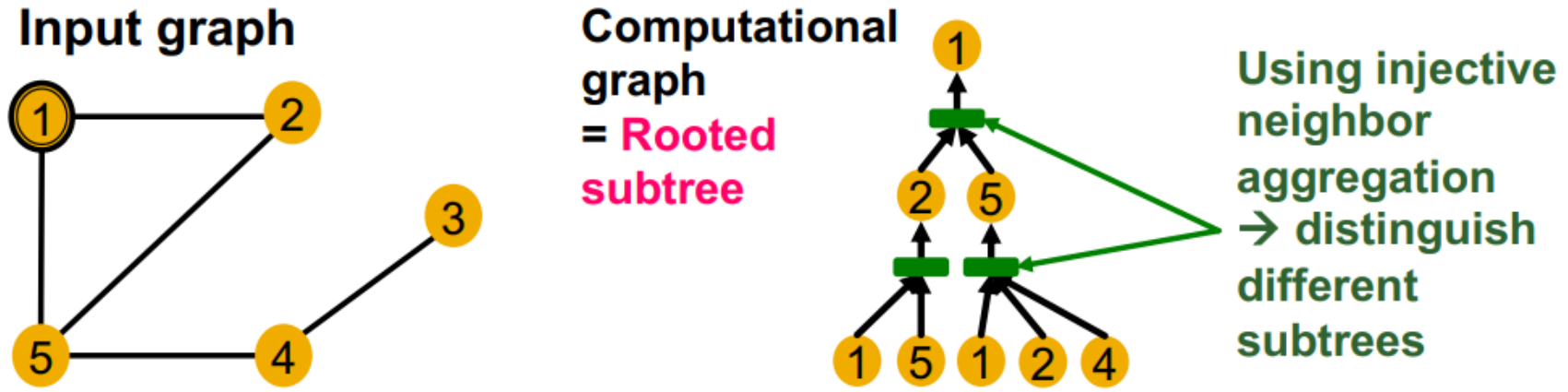
How expressive is a GNN?

- Most expressive GNN should map subtrees to the node embeddings injectively.



Summary:

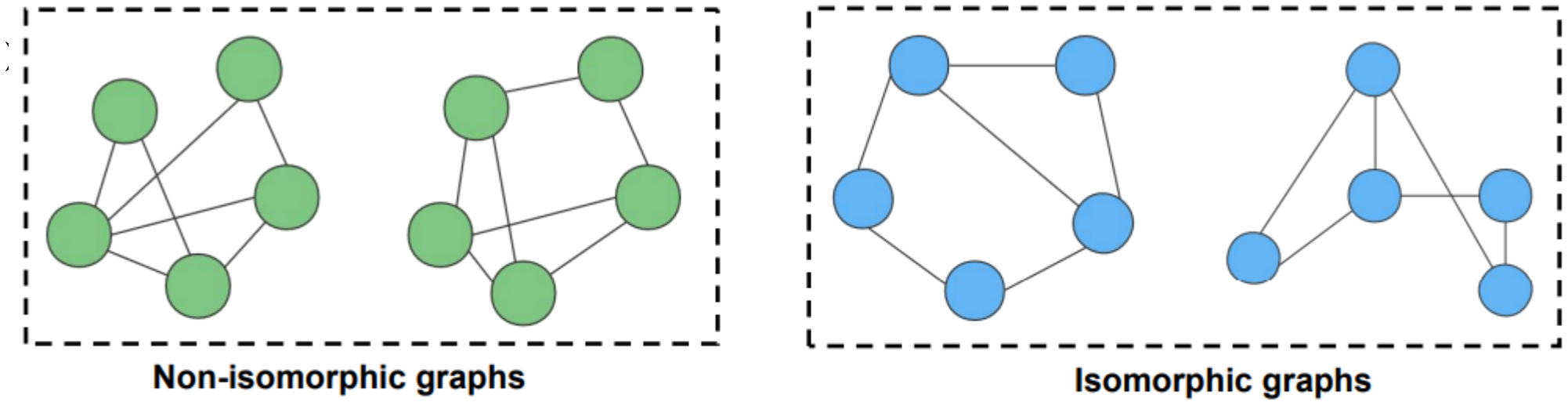
- To generate a node embedding, GNNs use a computational graph corresponding to a subtree rooted around each node.



- GNN can fully distinguish different subtree structures if every step of its neighbor aggregation is injective.

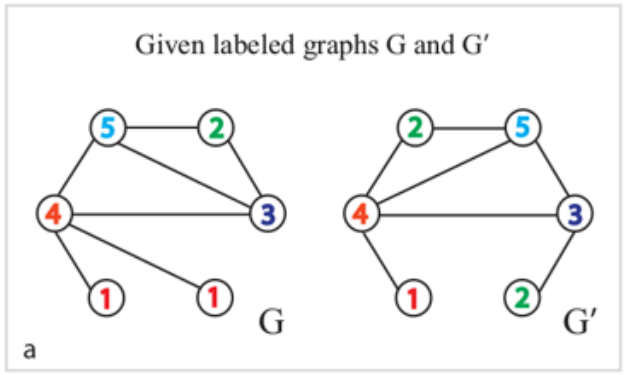
How to design the most powerful GNN?

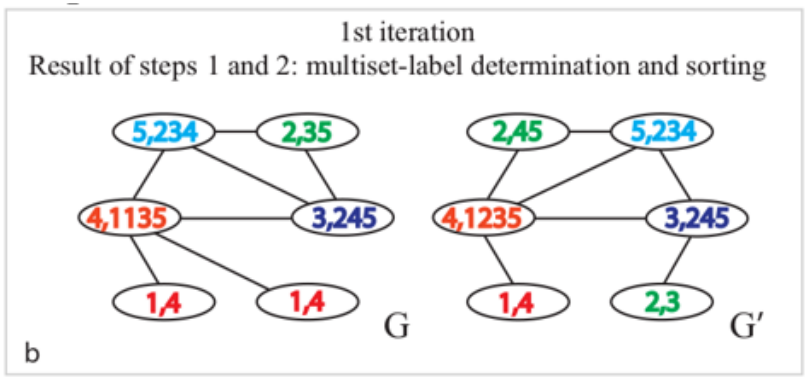
- A GNN model can distinguish the difference between two graphs

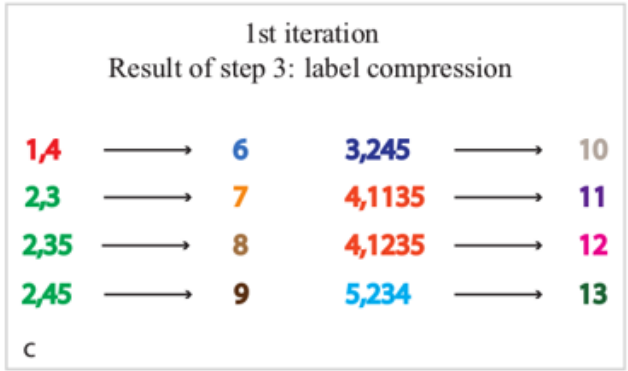


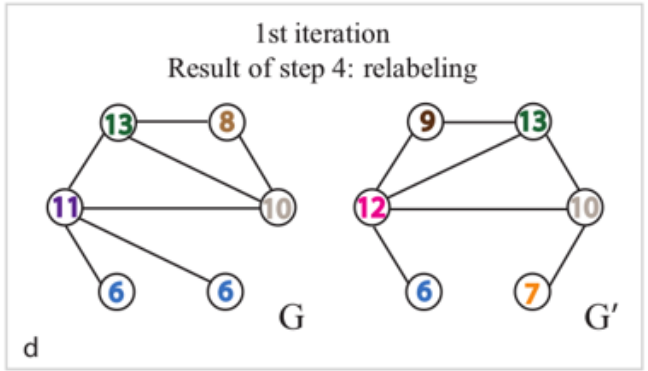
Understanding expressiveness: The theory of GNNs

- **Looking back to Isomorphism test**: The algorithm determines that two graphs are non-isomorphic if the lables of nodes between the two graphs differ in some iterations.
- Asks whether two graphs are topologically identical

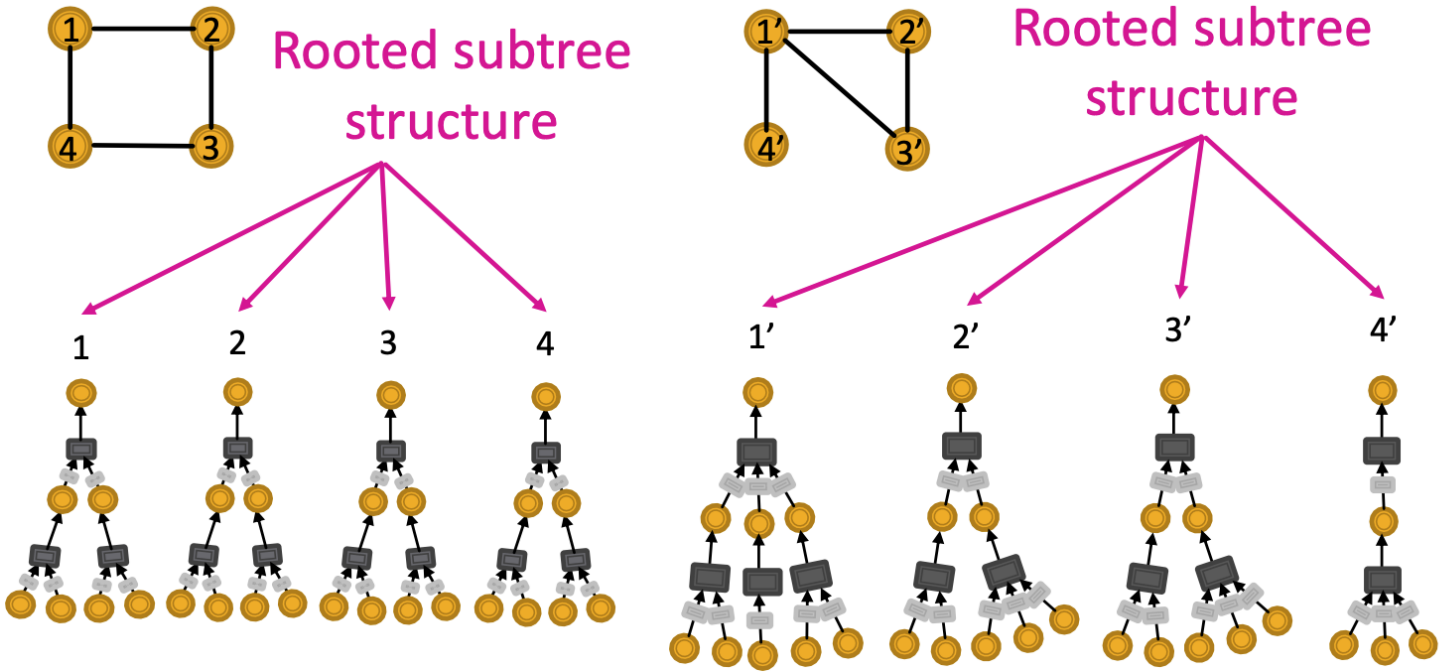






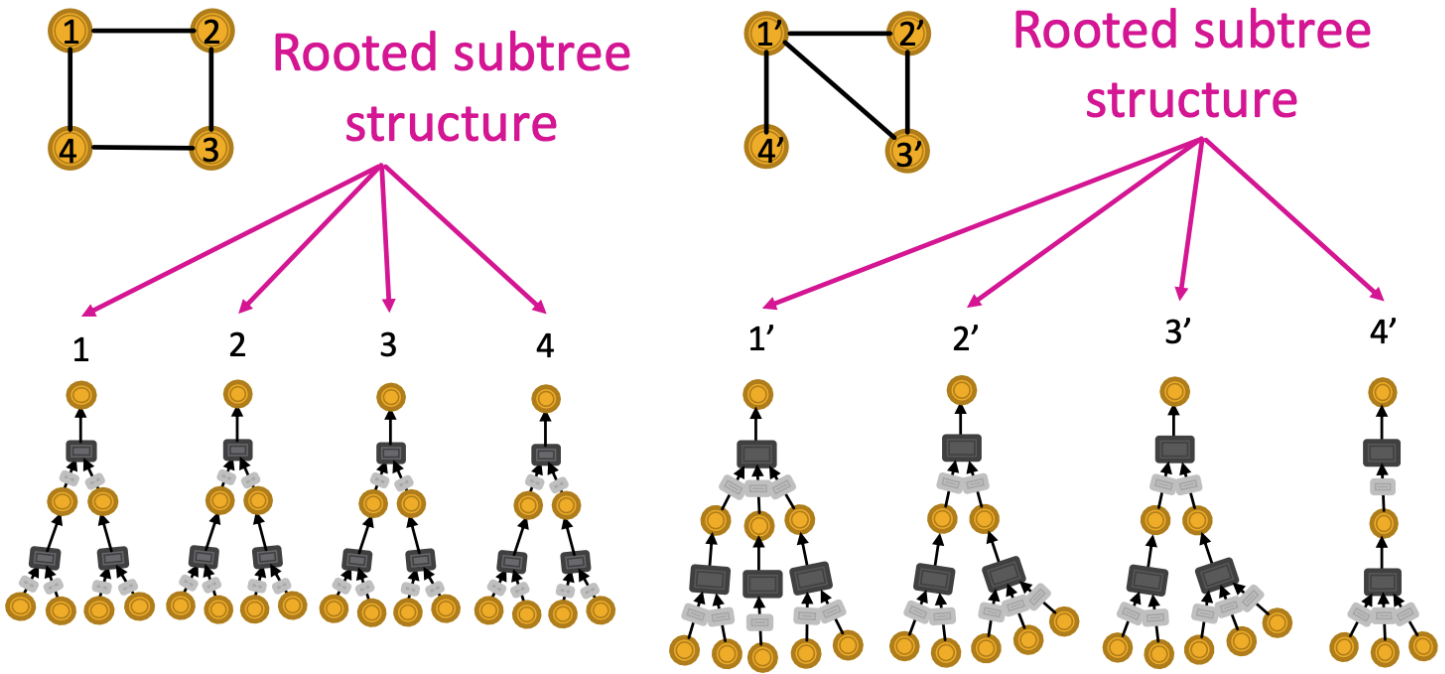


- A subtree pattern of height 2 rooted at the node 1.
- Node the repretitions of nodes in the unfolded subtree pattern on the right
  - The representation ability of GNN is studied by studying when GNN maps two nodes to the same feature representation.
  - The theoretical upper limit of the representation ability is that when the subtrees corresponding to the two nodes are the same, they are mapped to the same feature representation.



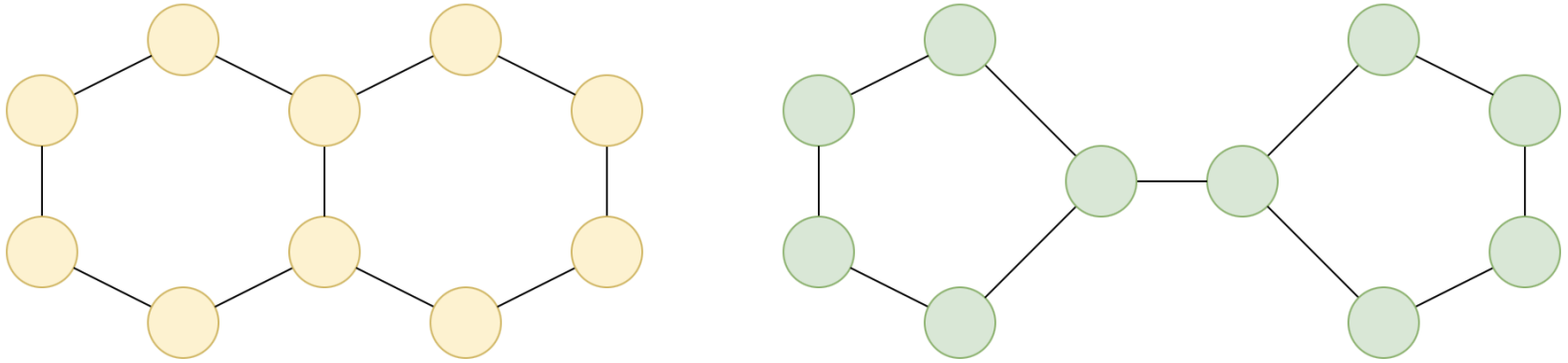
WL test and expressivity of MPNNs

- Represent the set of feature vectors of a given node's neighbors as a multiset
- A maximally powerful GNN would never map two different neighborhoods i.e., multisets of feature vectors, to the same representation.
  - Injective aggregation scheme
  - Isomorphic graphs have to be mapped to the same representation



Understanding Expressiveness

- Most GNNs fail to distinguish the difference between two graphs
- Examples of two graphs indistinguishable by the WL test.
  - They produce similar distributions through the iterations of colour refinement.
  - It’s catastrophic if datasets have graphs that exhibit similar properties and can’t be told apart.



Injective MPNN

- When every function in between is injective, the output function is injective as well:

- How can we build such model?

What makes GIN powerful?

- Consider a multiset $\mathcal{X}$, and ϵ to be any number
- Assume a function $g(c, X)$, where $c, X \in \mathcal{X}$, which maps each pair of inputs to a unique number
- There exists an f, such that for some function φ, g can be decomposed as follows:

$$g(c, X) = \varphi((1 + \epsilon) f(c) + \Sigma_{x \in X} f(x))$$

What makes GIN powerful?

- Model the $f^{(k+1)} \circ \phi^{(k)}$ with one MLP

$$h_v^{(k)} = \mathrm{MLP}^{(k)}((1+\epsilon^{(k)}).h_v^{(k-1)} + \Sigma_{u \in \mathcal{N}(v)} h_u^{(k-1)})$$

- ϵ can be a learnable parameter or a scalar

- **READOUT:**
  - More iterations gives better representational power
  - But less generalization
  - So GIN concatenates the embedding (information) from all layers

What makes GIN powerful?

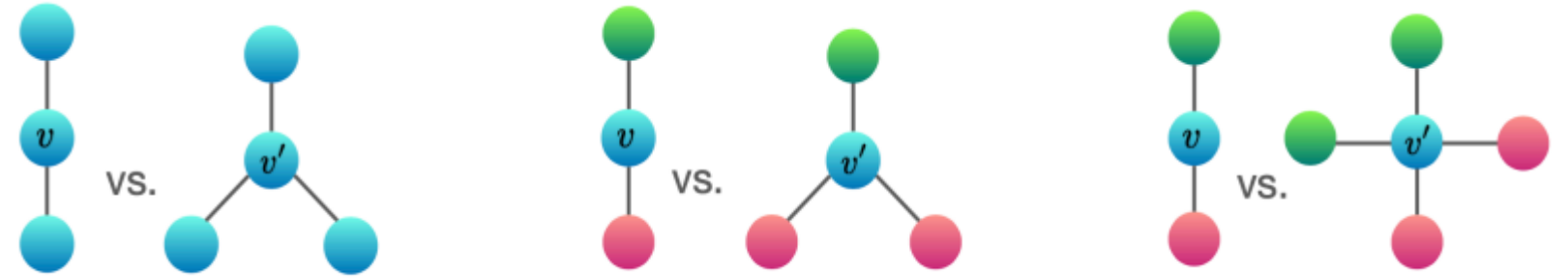
- 1-layer perceptron instead of MLP
- Linear mapping can map two multisets to the same representations
- Unlike models using MLPs, the 1-layer perceptron (even with the bias term) is not a universal approximator of multiset functions.

$$h_v^{(k)} = \mathrm{MLP}^{(k)}((1+\epsilon^{(k)}).h_v^{(k-1)} + \Sigma_{u\in\mathcal{N}(v)} h_u^{(k-1)})$$

Use an MLP with more than 1 layer

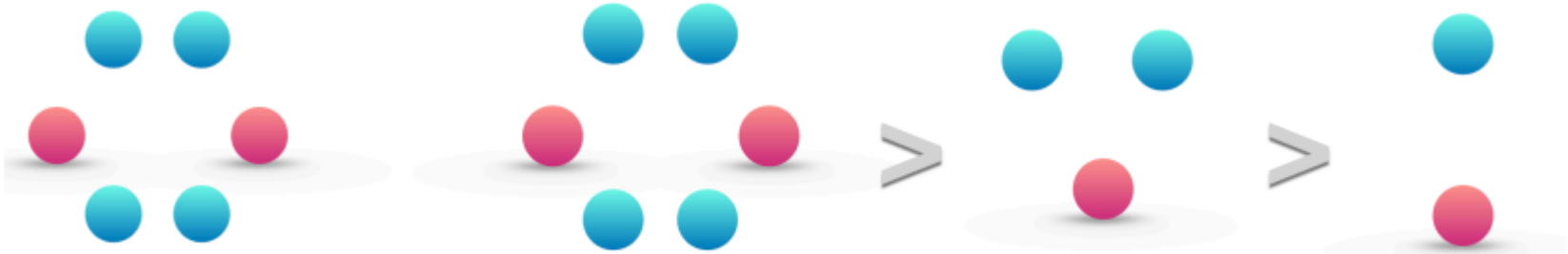
What makes GIN powerful?

- The mean captures the distribution (proportions) of elements in a multiset, but not the exact multiset
- Max can capture the skeleton



(a) Mean and Max both fail (b) Max fails (c) Mean and Max both fail

- Instead of sum in $h(X) = \Sigma_{x \in X} f(x)$, what if we use mean or max?

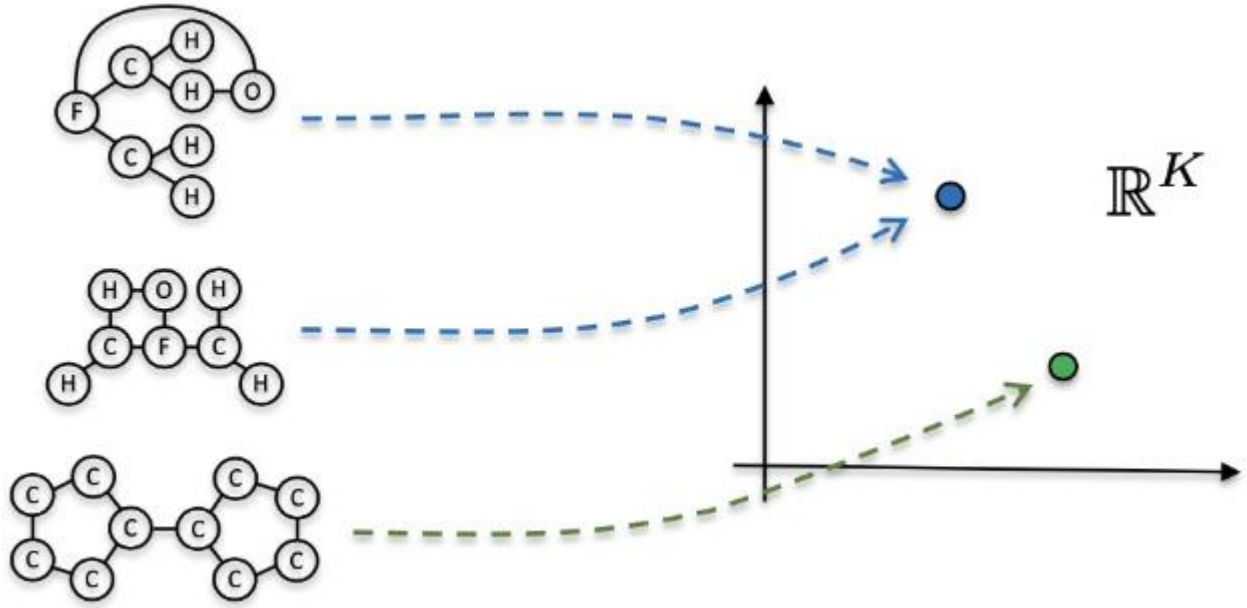


WL test and expressivity of GNNs

- GNNs are at most as powerful as the WL test in distinguishing graph structures
- Conditions on aggregation and the readout function to be as powerful as WL test

$$h_{\mathcal{G}} = \mathrm{MLP}\Big(\sum_{i \in V} h_i^L\Big) \in \mathbb{R}^K$$

Sum function



GIN in Torch Geometric

/ torch_geometric.nn / conv.GINConv

conv.GINConv

class **GINConv** (**nn**: Callable, **eps**: float = 0.0, **train_eps**: bool = False, ****kwargs**) [source]

Bases: `MessagePassing`

The graph isomorphism operator from the "How Powerful are Graph Neural Networks?" paper

$$\mathbf{x}_i' = h_{\mathbf{\Theta}}\left((1+\epsilon)\cdot\mathbf{x}_i + \sum_{j\in\mathcal{N}(i)}\mathbf{x}_j\right)$$

or

$$\mathbf{X}' = h_{\mathbf{\Theta}}\left((\mathbf{A}+(1+\epsilon)\cdot\mathbf{I})\cdot\mathbf{X}\right),$$

GIN: Sample code


```
61  class GIN(torch.nn.Module):
62          def __init__(self, num_features, num_classes, dim=16, drop=0.5):
63                  super(GIN, self).__init__()
64
65                  nn1 = Sequential(Linear(num_features, dim), ReLU(), Linear(dim, dim))
66                  self.conv1 = GINConv(nn1)
67                  self.bn1 = torch.nn.BatchNorm1d(dim)
68
69                  nn2 = Sequential(Linear(dim, dim), ReLU(), Linear(dim, num_classes))
70                  self.conv2 = GINConv(nn2)
71                  self.bn2 = torch.nn.BatchNorm1d(num_classes)
72
73                  self.drop = torch.nn.Dropout(p=drop)
74
75          def forward(self, x, edge_index):
76                  x = F.relu(self.conv1(x, edge_index))
77                  x = self.bn1(x)
78                  x = self.drop(x)
79                  x = F.relu(self.conv2(x, edge_index))
80                  x = self.bn2(x)
81                  return F.log_softmax(x, dim=1), x
82
```

- Let's try some simple GAT code in the sample code file

네트워크 과학 연구실
NETWORK SCIENCE LAB

가톨릭대학교
THE CATHOLIC UNIVERSITY OF KOREA